PARALLEL MCMC WITH GENERALIZED ELLIPTICAL SLICE SAMPLING

By Robert Nishihara, Iain Murray and Ryan P. Adams

Harvard University and University of Edinburgh

Probabilistic models are conceptually powerful tools for finding structure in data, but their practical effectiveness is often limited by our ability to perform inference in them. Exact inference is frequently intractable, so approximate inference is often performed using Markov chain Monte Carlo (MCMC). To achieve the best possible results from MCMC, we want to efficiently simulate many steps of a rapidly mixing Markov chain which leaves the target distribution invariant. Of particular interest in this regard is how to take advantage of multi-core computing to speed up MCMC-based inference, both to improve mixing and to distribute the computational load. In this paper, we present a parallelizable Markov chain Monte Carlo algorithm for efficiently sampling from continuous probability distributions that can take advantage of hundreds of cores. This method shares information between parallel Markov chains to build a scale-mixture of Gaussians approximation to the density function of the target distribution. We combine this approximation with a recent method known as elliptical slice sampling to create a Markov chain with no step-size parameters that can mix rapidly without requiring gradient or curvature computations.

1. Introduction. Probabilistic models are fundamental tools for machine learning, providing a coherent framework for finding structure in data. In the Bayesian formulation, learning is performed by computing a representation of the posterior distribution implied by the data. Unobserved quantities of interest can then be estimated as expectations of various functions under this posterior distribution.

These expectations typically correspond to high-dimensional integrals and sums, which are usually intractable for rich models. There is therefore significant interest in efficient methods for approximate inference that can rapidly estimate these expectations. In this paper, we examine Markov chain Monte Carlo (MCMC) methods for approximate inference, which estimate these quantities by simulating a Markov chain with the posterior as its equilibrium distribution. MCMC is often seen as a principled "gold standard" for inference, because (under mild conditions) its answers will be correct in the limit of simulations. However, in practice, MCMC often converges slowly and requires expert tuning. In this paper, we propose a new method to address these issues for continuous parameter spaces. We generalize the method of *elliptical slice sampling* (Murray et al., 2010) to build a new efficient method that: 1) mixes well in the presence of strong dependence, 2) does not require hand tuning, and 3) can take advantage of multiple computational cores operating in parallel. We discuss each of these in more detail below.

Many posterior distributions arising from real-world data have strong dependencies between variables. These dependencies can arise from correlations induced by the likelihood function, redundancy in the parameterization, or directly from the prior. One of the primary challenges for efficient Markov chain Monte Carlo is making large moves in directions that reflect the dependence structure. For example, if we imagine a long, thin region of high density, it is necessary to explore the length in order to reach equilibrium; however,

random-walk methods such as Metropolis–Hastings (MH) with spherical proposals can only diffuse as fast as the narrowest direction allows (Neal, 1995). More efficient methods such as Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2011; Girolami and Calderhead, 2011) avoid random walk behavior by introducing auxiliary 'momentum' variables. Hamiltonian methods require differentiable density functions and gradient computations.

In this work, we are able to make efficient long-range moves – even in the presence of dependence – by building an approximation to the target density that can be exploited by elliptical slice sampling. This approximation enables the algorithm to consider the general shape of the distribution without requiring gradient or curvature information. We construct the algorithm such that it is valid regardless of the quality of the approximation, preserving the guarantees of approximate inference by MCMC.

One of the limitations of MCMC in practice is that it is often difficult for non-experts to apply. This difficulty stems from the fact that it can be challenging to tune the Markov transition operators so that they mix well. For example, in Metropolis–Hastings, one must come up with appropriate proposal distributions. In Hamiltonian Monte Carlo, one must choose the number of steps and the step size in the simulation of dynamics. For probabilistic machine learning methods to be widely applicable, it is necessary to develop black-box methods for approximate inference that do not require extensive hand tuning. Some recent attempts have been made in the area of adaptive MCMC (Roberts and Rosenthal, 2006; Haario et al., 2005), but these are only theoretically understood for a relatively narrow class of transition operators (e.g. not Hamiltonian Monte Carlo). Here we propose a method based on slice sampling (Neal, 2003), which uses a local search to find an acceptable point, and avoid potential issues with convergence under adaptation.

In all aspects of machine learning, a significant challenge is exploiting a computational landscape that is evolving toward parallelism over single-core speed. When considering parallel approaches to MCMC, we can readily identify two ends of a spectrum of possible solutions. At one extreme, we could run a large number of independent Markov chains in parallel (Rosenthal, 2000; Bradford and Thomas, 1996). This will have the benefit of providing more samples and increasing the accuracy of the end result, however it will do nothing to speed up the convergence or the mixing of each individual chain. The parallel algorithm will run up against the same limitations faced by the non-parallel version. At another extreme, we could develop a single-chain MCMC algorithm which parallelizes the individual Markov transitions in a problem-specific way (Suchard and Rambaut, 2009; Suchard et al., 2010; Tarlow et al., 2012). For instance, if the likelihood is expensive and consists of many factors, the factors can potentially be computed in parallel. Alternatively, some Markov chain transition operators can make use of multiple parallel proposals to increase their efficiency, such as multiple-try Metropolis–Hastings (Liu et al., 2000).

We propose an intermediate algorithm to make effective use of parallelism. By sharing information between the chains, our method is able to mix faster than the naïve approach of running independent chains. However, we do not require fine-grained control over parallel execution, as would be necessary for the single-chain method. Nevertheless, if such local parallelism is possible, our sampler can take advantage of it. Our general objective is a black-box approach that mixes well with multiple cores but does not require the user to build in parallelism.

The structure of the paper is as follows. In Section 2, we review slice sampling (Neal, 2003) and elliptical slice sampling (Murray et al., 2010). In Section 3, we show how an elliptical approximation to the target distribution enables us to generalize elliptical slice sampling to continuous distributions. In Section 4, we describe a natural way to use parallelism to

dynamically construct the desired approximation. In section 5, we discuss related work. In Section 6, we evaluate our new approach against other comparable methods on several typical modeling problems.

2. Background. Throughout this paper, we will use $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to denote the density function of a Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ evaluated at a point $\mathbf{x} \in \mathbb{R}^d$. We will use $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to refer to the distribution itself. Analogous notation will be used for other distributions. Throughout, we shall assume that we wish to draw samples from a probability distribution over $\mathbb{R}^d$ whose density function is π. We sometimes refer to the distribution itself as π.

The objective of Markov chain Monte Carlo is to formulate transition operators that can be easily simulated, that leave π invariant, and that are ergodic. Classical examples of MCMC are Metropolis–Hastings (Metropolis et al., 1953; Hastings, 1970) and Gibbs Sampling (Geman and Geman, 1984). For general overviews of MCMC, see Tierney (1994); Andrieu et al. (2003); Brooks et al. (2011). Simulating such a transition operator will, in the limit, produce samples from π, and these can be used to compute expectations under π. In general, we do not have access to π itself; we may only have access to a scalar multiple of π. However, none of the algorithms that we describe require access to the normalization constant, and so we will abuse notation somewhat and refer to the unnormalized density as π.

2.1. *Slice Sampling.* Slice sampling (Neal, 2003) is a Markov chain Monte Carlo algorithm with an adaptive step size. It is an auxiliary-variable method, which relies on the observation that sampling π is equivalent to sampling the uniform distribution over the set $S = \{(\mathbf{x}, y) \mid 0 \leq y \leq \pi(\mathbf{x})\}$ and disregarding the y coordinate. Slice sampling accomplishes this by alternately updating $\mathbf{x}$ and y so as to leave invariant the distributions $p(\mathbf{x} \,|\, y)$ and $p(y \,|\, \mathbf{x})$ respectively. The key insight of slice sampling is that sampling from these conditionals (which correspond to "slices" under the density function) is potentially much easier than sampling directly from π.

Updating y according to $p(y \,|\, \mathbf{x})$ is trivial. The new value of y is drawn uniformly from the interval $[0, \pi(\mathbf{x})]$. There are different ways of updating $\mathbf{x}$. The objective is to draw uniformly from among the "slice" $\{\mathbf{x} \mid \pi(\mathbf{x}) \geq y\}$. Typically, this is done by defining a transition operator that leaves the uniform distribution on the slice invariant. Neal (2003) describes such a transition operator: first, choose a direction in which to search, then place an interval around the current state, expand it as necessary, and shrink it until an acceptable point is found. Several procedures have been proposed for the expansion and contraction phases.

Less clear is how to choose an efficient direction in which to search. There are two approaches that are widely used. First, one could choose a direction uniformly at random from all possible directions (MacKay, 2003). Second, one could choose a direction uniformly at random from the d coordinate directions. We consider both of these implementations later, and we refer to them as *random-direction slice sampling* (RDSS) and *coordinate-wise slice sampling* (CWSS) respectively. In principle, any distribution over directions can be used as long as detailed balance is satisfied, but it is unclear what form this distribution should take. The choice of direction has a significant impact on how rapidly mixing occurs. In the remainder of the paper, we describe how slice sampling can be modified so that candidate points are chosen to reflect the structure of the target distribution.

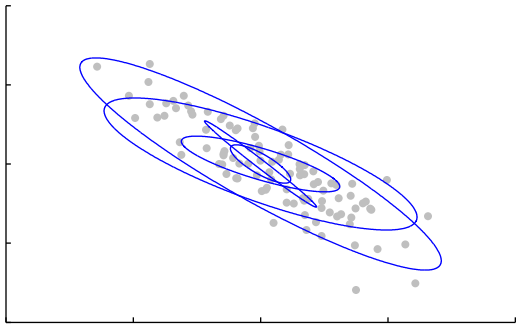


Fig 1: Background points are drawn independently from a probability distribution, and five ellipses are created by elliptical slice sampling.

2.2. *Elliptical Slice Sampling.* Elliptical slice sampling (Murray et al., 2010) is an MCMC algorithm designed to sample from posteriors over latent variables of the form

$$\pi(\mathbf{x}) \propto L(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1}$$

where L is a likelihood function, and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian prior. Such models, often called *latent Gaussian models*, arise frequently from Gaussian processes and Gaussian Markov random fields. Elliptical slice sampling takes advantage of the structure induced by the Gaussian prior to mix rapidly even when the covariance induces strong dependence. The method is easier to apply than most MCMC algorithms because it has no free tuning parameters.

Elliptical slice sampling takes advantage of a convenient invariance property of the multivariate Gaussian. Namely, if $\mathbf{x}$ and $\boldsymbol{\nu}$ are independent draws from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the linear combination

$$\mathbf{x}' = (\mathbf{x} - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{\nu} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu} \tag{2}$$

is also (marginally) distributed according to $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for any $\theta \in [0, 2\pi]$. Note that $\mathbf{x}'$ is nevertheless correlated with $\mathbf{x}$ and $\boldsymbol{\nu}$. This correlation has been previously used to make perturbative Metropolis–Hastings proposals in latent Gaussian models (Neal, 1998; Adams et al., 2009), but elliptical slice sampling uses it as the basis for a rejection-free method.

The elliptical slice sampling transition operator considers the locus of points defined by varying θ in Equation 2. This locus is an ellipse which passes through the current state $\mathbf{x}$ as well as through the auxiliary variable $\boldsymbol{\nu}$. Although all values of θ leave the prior invariant, some of these are excluded by the likelihood. Given a random ellipse induced by $\boldsymbol{\nu}$, we can slice sample $\theta \in [0, 2\pi]$ to choose the next point based purely on the likelihood term. The advantage of this procedure is that the ellipses will necessarily reflect the dependence induced by strong Gaussian priors and that the user does not have to choose a step size. Figure 1 depicts random ellipses produced by elliptical slice sampling superimposed on background points from some target distribution. This diagram illustrates the idea that the ellipses produced by elliptical slice sampling reflect the structure of the distribution.

3. Generalized Elliptical Slice Sampling. In this section, we generalize elliptical slice sampling to handle arbitrary continuous distributions. We refer to this algorithm as *generalized elliptical slice sampling* (GESS). In this section, our target distribution will be a continuous distribution over $\mathbb{R}^d$ with density π. In practice, π need not be normalized.

Algorithm 1 Elliptical Slice Sampling Update

Input: Current state $\mathbf{x}$, Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, log-likelihood function $\log L$
Output: New state $\mathbf{x}'$, with stationary distribution proportional to $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})L(\mathbf{x})$

1: $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ▷ Choose ellipse
2: $u \sim \text{Uniform}[0, 1]$
3: $\log y \leftarrow \log L(\mathbf{x}) + \log u$ ▷ Set log-likelihood threshold
4: $\theta \sim \text{Uniform}[0, 2\pi]$ ▷ Draw an initial proposal
5: $[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi, \theta]$ ▷ Define a bracket
6: $\mathbf{x}' \leftarrow (\mathbf{x} - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{\nu} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}$
7: **if** $\log L(\mathbf{x}') > \log y$ **then**
8: **return** $\mathbf{x}'$ ▷ Accept
9: **else** ▷ Shrink the bracket and try a new point
10: **if** $\theta < 0$ **then**
11: $\theta_{\min} \leftarrow \theta$
12: **else**
13: $\theta_{\max} \leftarrow \theta$
14: $\theta \sim \text{Uniform}[\theta_{\min}, \theta_{\max}]$
15: **goto** 6

Our objective is to reframe our target distribution so that it can be efficiently sampled with elliptical slice sampling. One possible approach is to put π in the form of Equation (1) by letting

$$L(\mathbf{x}) = \frac{\pi(\mathbf{x})}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}, \tag{3}$$

for some choice of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Note that $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is an approximation rather than a prior and that L is not a likelihood function, but since the equation has the correct form, this representation enables us to use elliptical slice sampling.

Applying elliptical slice sampling in this manner will produce a correct algorithm in the limit, but it may mix slowly in practice. Difficulties arise when the target distribution has much heavier tails than does the approximation. In such a situation, $L(\mathbf{x})$ will increase rapidly as $\mathbf{x}$ moves away from the mean of the approximation. To illustrate this phenomenon, we use this approach with different approximations to draw samples from a Gaussian in one-dimension with zero mean and unit variance. Trace plots are shown in Figure 2. The subplot corresponding to variance 0.01 illustrates the problem. Since the likelihood function explodes as $|\mathbf{x}|$ gets large, the Markov chain is unlikely to move back toward the origin. On the other hand, the size of the ellipse is limited by a draw from the Gaussian approximation, which has low variance in this case, so the Markov chain is also unlikely to move away from the origin. The result is that the Markov chain sometimes gets stuck. In the subplot corresponding to variance 0.01, this occurs between iterations 400 and 630.

We resolve this dilemma by extending elliptical slice sampling to distributions whose densities have the more general form of a likelihood function multiplied by a scale-location mixture of Gaussians:

$$\pi(\mathbf{x}) \propto L(\mathbf{x}) \int \phi(s)\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s))\, \mathrm{d}s, \tag{4}$$

where s is some auxiliary parameter with density function ϕ. Here, ϕ can be chosen in a problem-specific way, and $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are taken to be functions of s. We observe that any continuous density can be approximated to an arbitrary precision by a sufficiently rich scale-location mixture of Gaussians. This construction then allows any residual error between π and the approximation to be fixed by L.

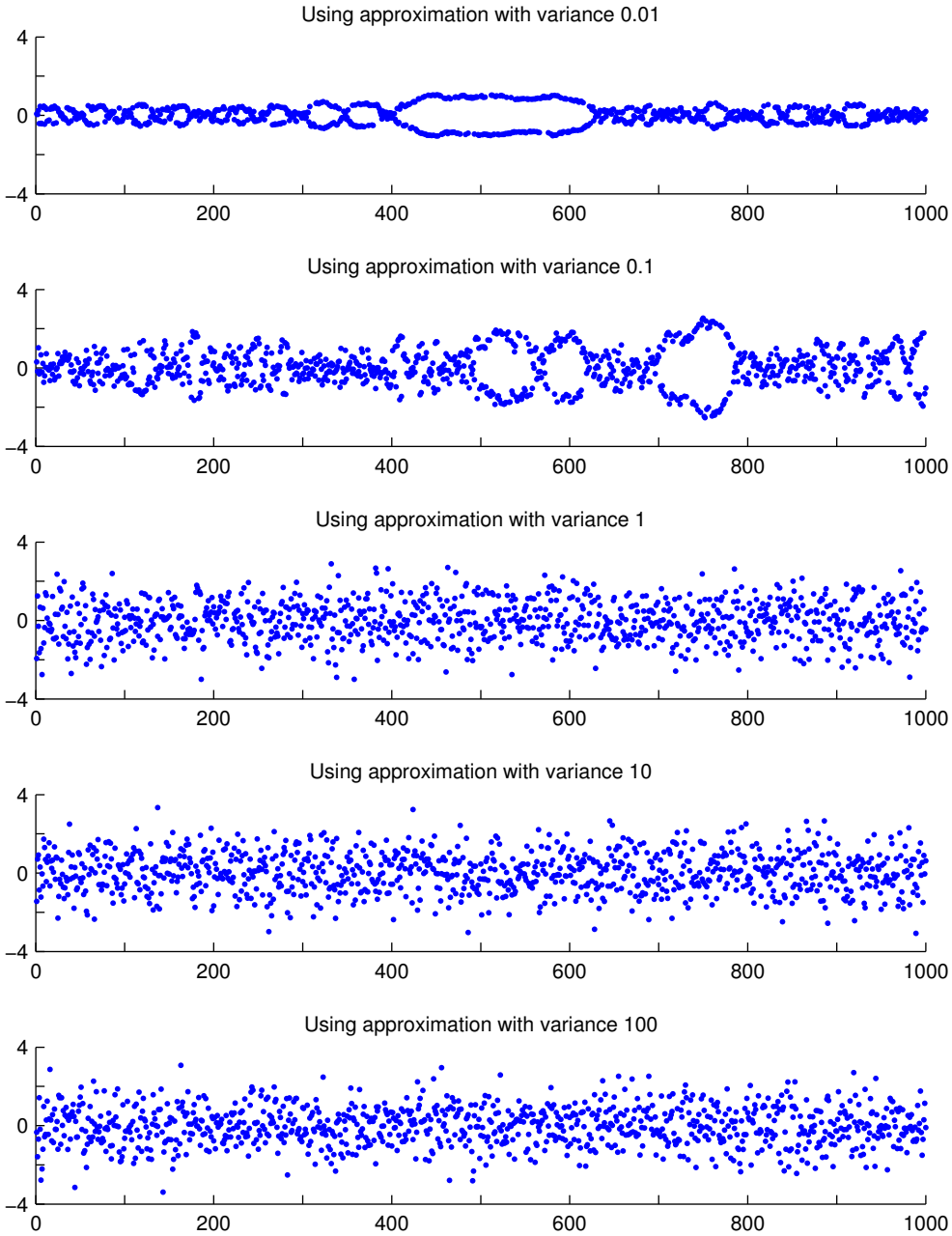


Fig 2: Samples are drawn from a Gaussian with zero mean and unit variance using elliptical slice sampling with various Gaussian approximations. These trace plots show how sampling behavior depends on how heavy the tails of the approximation are relative to how heavy the tails of the target distribution are. We plot one of every ten samples.

Through Equation (4), we can view $\pi(\mathbf{x})$ as the marginal density of an augmented joint distribution

$$p(\mathbf{x}, s) = L(\mathbf{x})\,\phi(s)\,\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s)). \tag{5}$$

Therefore, to sample π, it suffices to sample $\mathbf{x}$ and s jointly and then to disregard the s coordinate. We update these components separately according to

$$p(\mathbf{x} \mid s) \propto L(\mathbf{x})\,\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s)) \tag{6}$$

and

$$p(s \mid \mathbf{x}) \propto \phi(s)\,\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}(s), \boldsymbol{\Sigma}(s)). \tag{7}$$

Equation (6) has the correct form for elliptical slice sampling and can be updated according to Algorithm 1. Equation (7) can be updated using any valid Markov transition operator.

We now focus on a particular case in which the update given by Equation (7) is easy to simulate and in which we can make the tails as heavy as we desire, so as to control the behavior of L. A simple and convenient choice is for the scale-location mixture to yield a multivariate t distribution with degrees-of-freedom parameter ν:

$$\mathcal{T}_\nu(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_0^\infty \mathrm{IG}(s; \tfrac{\nu}{2}, \tfrac{\nu}{2})\,\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, s\boldsymbol{\Sigma})\,\mathrm{d}s, \tag{8}$$

where ϕ becomes the density function of an inverse-gamma distribution:

$$\mathrm{IG}(s; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} s^{-\alpha-1} e^{-\beta/s}. \tag{9}$$

Here s is a scale parameter. Now, in the update $p(s \mid \mathbf{x})$, we observe that the inverse-gamma distribution is a conjugate prior, so

$$p(s \mid \mathbf{x}) = \mathrm{IG}(s; \alpha, \beta) \tag{10}$$

with parameters

$$\alpha = \frac{d + \nu}{2} \quad \text{and} \tag{11}$$

$$\beta = \frac{1}{2}(\nu + (\mathbf{x} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})). \tag{12}$$

We can draw independent samples from this distribution.

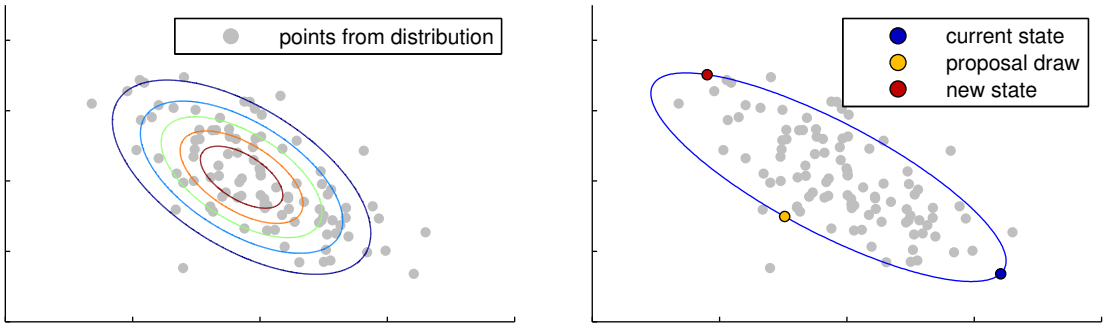
Combining these update steps, we define the transition operator $S(\mathbf{x} \to \mathbf{x}'; \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to be the one which draws $s \sim \mathrm{IG}(s; \alpha, \beta)$ (with α and β as described in Equations (11) and (12)) and then uses elliptical slice sampling to update $\mathbf{x}$ as described in Algorithm 1. From the above discussion, it follows that the stationary distribution of $S(\mathbf{x} \to \mathbf{x}'; \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is π. Figure 3 illustrates this transition operator.

4. Building the Approximation with Parallelism. Up to this point, we have not described how to choose the multivariate t parameters ν, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$. These choices can be made in many ways. For instance, we may choose the maximum likelihood parameters given samples collected during a burn in period, we may build a Laplace approximation to the mode of the distribution, or we may use variational approaches. Note that this algorithm is

Algorithm 2 Generalized Elliptical Slice Sampling Update

Input: Current state $\mathbf{x}$, multivariate t parameters $\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}$, dimension d, a routine ESS that performs an elliptical slice sampling update
Output: New state $\mathbf{x}'$

$\alpha \leftarrow \frac{d+\nu}{2}$
$\beta \leftarrow \frac{1}{2}(\nu + (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$
$s \sim \mathrm{IG}(\alpha, \beta)$
$\log L \leftarrow \log \pi - \log \mathcal{T}$ ▷ $\mathcal{T}$ is the density of a multivariate t with parameters $\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
$\mathbf{x}' \leftarrow \mathrm{ESS}(\mathbf{x}, \boldsymbol{\mu}, s\boldsymbol{\Sigma}, \log L)$



(a) Multivariate t contours (b) Example GESS update

Fig 3: The gray points were drawn independently from a two-dimensional Gaussian to show the rough shape of the distribution. **(a)** Shows the contours of a multivariate t approximation to this distribution. **(b)** Shows a sample update step using the transition operator $S(\mathbf{x} \to \mathbf{x}'; \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The blue point represents the current state, the yellow point is drawn from the t approximation defining an ellipse, and the red point corresponding to the new state is sampled from the given ellipse.

valid regardless of the particular choice we make here. In this section, we discuss a convenient way to use parallelism to dynamically choose these parameters. This method creates a large number of parallel Markov chains, each with π as its stationary distribution, and it divides them into two groups. The need for two groups of Markov chains is not immediately obvious, so we motivate our approach by first discussing two simpler algorithms that fail in different ways.

4.1. *Naïve Approaches.* We begin with a collection of K parallel Markov chains. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ denote the current states of the Markov chains. We observe that $\mathcal{X}$ may contain a lot of information about the shape of the target distribution. We would like to define a transition operator $Q(\mathcal{X} \to \mathcal{X}')$ that utilizes this information to intelligently choose the multivariate t parameters ν, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ and then uses these parameters to update each $\mathbf{x}_k$ via generalized elliptical slice sampling. Additionally, we would like Q to have two properties. First, each $\mathbf{x}_k$ should have marginal distribution π. Second, we should be able to parallelize the update of $\mathcal{X}$ over K cores.

Here we describe two simple approaches for parallelizing generalized elliptical slice sampling, each of which lacks one of the desired properties. The first approach begins with K parallel Markov chains, and it requires a procedure for choosing the multivariate t parameters given $\mathcal{X}$. In this setup, Q uses this procedure to determine the multivariate t parameters

from $\mathcal{X}$ and then applies $S(\mathbf{x} \to \mathbf{x}'; \nu_{\mathcal{X}}, \boldsymbol{\mu}_{\mathcal{X}}, \boldsymbol{\Sigma}_{\mathcal{X}})$ to each $\mathbf{x}_k$ individually. These updates can be performed in parallel, but we lose the guarantee that the variables $\mathbf{x}_k$ have the correct marginal distributions.

A second approach creates a valid MCMC method by including the multivariate t parameters in a joint distribution

$$p(\mathcal{X}, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathcal{X}) \left[\prod_{k=1}^{K} \pi(\mathbf{x}_k) \right]. \tag{13}$$

Note that in Equation (13), each $\mathbf{x}_k$ has marginal distribution π. We sample this joint distribution by alternately sampling $p(\mathcal{X} \,|\, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathcal{X})$. We would like to be able to parallelize the sampler that draws from the first of these conditional distributions. We can write

$$p(\mathcal{X} \,|\, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathcal{X}) \prod_{k=1}^{K} \pi(\mathbf{x}_k) \tag{14}$$

Ideally, we would like to update the collection $\mathcal{X}$ by updating each $\mathbf{x}_k$ in parallel. However, we cannot easily parallelize the update in this formulation because of the factor of $p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathcal{X})$, which introduces dependence between the chains.

4.2. *The Two-Group Approach.* Our proposed method creates a transition operator Q that satisfies both of the desired properties. That is, each $\mathbf{x}_k$ has marginal distribution π, and the update can be efficiently parallelized. This method circumvents the problems of the previous approaches by maintaining two groups of Markov chains and using each group to choose multivariate t parameters to update the other group. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{K_1}\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{K_2}\}$ denote the states of the Markov chains in these two groups. The stationary distribution of the collection is

$$\Pi(\mathcal{X}, \mathcal{Y}) = \Pi_1(\mathcal{X})\Pi_2(\mathcal{Y}) = \left[\prod_{k=1}^{K_1} \pi(\mathbf{x}_k) \right] \left[\prod_{k=1}^{K_2} \pi(\mathbf{y}_k) \right]. \tag{15}$$

By simulating a Markov chain which leaves this product distribution invariant, this method generates samples from the target distribution. Our Markov chain is based on a transition operator, Q, defined in two parts. The first part of the transition operator, Q_1, uses $\mathcal{Y}$ to determine parameters $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$. It then uses these parameters to update $\mathcal{X}$. The second part of the transition operator, Q_2, uses $\mathcal{X}$ to determine parameters $\nu_{\mathcal{X}}$, $\boldsymbol{\mu}_{\mathcal{X}}$, and $\boldsymbol{\Sigma}_{\mathcal{X}}$. It then uses these parameters to update $\mathcal{Y}$. The transition operator Q is the composition of Q_1 and Q_2.

In order to make these descriptions more precise, we define Q_1 as follows. First, we specify a procedure for choosing the multivariate t parameters given the population $\mathcal{Y}$. Our implementation is deterministic, but the procedure can be made stochastic if desired. We use an extension of the expectation-maximization algorithm (Liu and Rubin, 1995) to choose the maximum-likelihood multivariate t parameters given the data $\mathcal{Y}$. More concretely, we choose

$$\nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}} = \arg\max_{\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}} \prod_{k=1}^{K_2} \mathcal{T}_{\nu}(\mathbf{y}_k \,;\, \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{16}$$

Algorithm 3 Building the Approximation Using Parallelism

Input: Two groups of states $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{k_1}\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{k_2}\}$, a subroutine FIT-MVT which takes data and returns the maximum-likelihood t parameters, a subroutine GESS which performs a generalized elliptical slice sampling update
Output: Updated groups $\mathcal{X}'$ and $\mathcal{Y}'$
1: $\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow$ FIT-MVT($\mathcal{Y}$)
2: **for all** $\mathbf{x}_k \in \mathcal{X}$ **do**
3: $\quad \mathbf{x}'_k = \text{GESS}(\mathbf{x}_k, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
4: $\mathcal{X}' \leftarrow \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{k_1}\}$
5: $\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow$ FIT-MVT($\mathcal{X}'$)
6: **for all** $\mathbf{y}_k \in \mathcal{Y}$ **do**
7: $\quad \mathbf{y}'_k = \text{GESS}(\mathbf{y}_k, \nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
8: $\mathcal{Y}' \leftarrow \{\mathbf{y}'_1, \ldots, \mathbf{y}'_{k_2}\}$

In the case where the number of chains in the collection $\mathcal{Y}$ is less than or close to the dimension of the distribution, the particular algorithm that we use to choose the parameters (Liu and Rubin, 1995) may not converge quickly. In this situation, we can perform a regularized estimate of the parameters by padding the collection $\mathcal{Y}$ with additional data drawn from a spherical Gaussian. This corresponds to performing a penalized maximum-likelihood estimate. After choosing $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$ in this manner, we update $\mathcal{X}$ by applying $S(\mathbf{x} \to \mathbf{x}'; \nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}})$ to each $\mathbf{x}_k \in \mathcal{X}$. The operator Q_2 is defined analogously.

4.3. *Correctness.* To establish the correctness of our algorithm, we treat the collection of Markov chains as a single aggregate Markov chain, and we show that this aggregate Markov chain with transition operator Q correctly samples from the product distribution Π.

We wish to show that Q_1 and Q_2 preserve the stationary distributions Π_1 and Π_2 respectively. As the two cases are identical, we consider only the first. We have

$$\int \Pi_1(\mathcal{X})\, Q_1(\mathcal{X} \to \mathcal{X}')\, \mathrm{d}\mathcal{X} = \int \Pi_1(\mathcal{X})\, Q_1(\mathcal{X} \to \mathcal{X}' \,|\, \nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}})\, \mathrm{d}\mathcal{X} \tag{17}$$

$$= \prod_{k=1}^{K_1} \left[\int \pi(\mathbf{x}_k)\, S(\mathbf{x}_k \to \mathbf{x}'_k; \nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}})\, \mathrm{d}\mathbf{x}_k \right] \tag{18}$$

$$= \Pi_1(\mathcal{X}'). \tag{19}$$

The last equality uses the fact that π is the stationary distribution of $S(\mathbf{x} \to \mathbf{x}'; \nu_{\mathcal{Y}}, \boldsymbol{\mu}_{\mathcal{Y}}, \boldsymbol{\Sigma}_{\mathcal{Y}})$. This demonstrates that Q leaves the desired product distribution invariant.

Within a single chain, elliptical slice sampling has a nonzero probability of transitioning to any region that has nonzero probability under the posterior, as described by Murray et al. (2010). The transition operator Q updates the chains independently of one another. Therefore Q has a nonzero probability of transitioning to any region that has nonzero probability under the product distribution. It follows that the transition operator is both irreducible and aperiodic. These conditions together ensure that this Markov transition operator has a unique stationary distribution, namely Π, and that the distribution over the state of the Markov chain created from this transition operator will converge to this stationary distribution (Roberts and Rosenthal, 2004). It follows that, in the limit, samples derived from the repeated application of Q will be drawn from the desired distribution.

4.4. *Cost and Complexity.* There is a cost to the construction of the multivariate t approximation. Although the user has some flexibility in the choice of t parameters, we fit

them with an iterative algorithm described by Liu and Rubin (1995). Let d be the dimension of the distribution and let k be the number of parallel chains. Then the complexity of each iteration is $O(d^3k)$. Empirically, the algorithm appears to converge in a small number of iterations when the number of parallel Markov chains in each group exceeds the dimension of the distribution. As described in the next section, this cost can be amortized by reusing the same approximation for multiple updates.

An additional concern is the overhead from sharing information between chains. The chains must communicate in order to build a multivariate t approximation, and so the updates must be synchronized. Since elliptical slice sampling requires a variable amount of time, updating the different chains will take different amounts of time, and the faster chains may end up waiting for the slower ones. We can mitigate this cost by performing multiple updates between such periods of information sharing. In this manner, we can perform as much computation as we want between synchronizations without compromising the validity of the algorithm. As we increase the number of updates performed between synchronizations, the percentage of wall-clock time spent waiting will diminish.

4.5. *Reusing the Approximation.* Here we explain that reusing the same approximation is valid, at least when our procedure for determining the multivariate t approximation is deterministic. To illustrate this point, let the transition operators Q_1 and Q_2 be defined as before. In our description of the algorithm, we defined the transition operator $Q = Q_2Q_1$. However, both Q_1 and Q_2 preserve the desired product distribution, so we may use any transition operator of the form $Q = Q_2^{r_2}Q_1^{r_1}$, where this notation indicates that we first apply Q_1 for r_1 rounds and then we apply Q_2 for r_2 rounds. As long as $r_2, r_1 \geq 1$, the composite transition operator has the proper behavior. This formulation makes apparent the benefit of using a deterministic algorithm for choosing the multivariate t parameters. When we apply Q_1 multiple times in a row, the states $\mathcal{Y}$ do not change, so if Q_1 computes $\nu_{\mathcal{Y}}$, $\boldsymbol{\mu}_{\mathcal{Y}}$, and $\boldsymbol{\Sigma}_{\mathcal{Y}}$ deterministically from $\mathcal{Y}$, then we need only compute these values once.

Our algorithm maintains two collections of Markov chains, one of which will always be idle. Therefore, each collection can take advantage of all available cores. Given K cores, it makes sense to use two collections of K Markov chains. In general, it is a good idea to sample equally from both collections so that the chains in both collections burn in. However, it is not absolutely necessary to give the groups equal roles. In our experiments, we treat the first group as an auxiliary collection to aid in the building of approximations, and we use the second group to do the sampling. We do this in order to make our algorithm comparable to other parallel algorithms, which sample using K Markov chains, for the purpose of our experiments in Section 6.

5. Related Work. Our work uses updates on a product distribution in the style of Adaptive Direction Sampling (Gilks et al., 1994), which has inspired a large literature of related methods. The closest research to our work makes use of slice-sampling based updates of product distributions along straight-line directions chosen by sampling pairs of points (MacKay, 2003; Ter Braak, 2006). The work on elliptical slice sampling suggests that in high dimensions larger steps can be taken along curved trajectories, given an appropriate Gaussian fit. Using closed ellipses also removes the need to set an initial step size.

The recent affine invariant ensemble sampler (Goodman and Weare, 2010) also uses Gaussian fits to a population, in that case to make Metropolis proposals. Our work differs by using a scale-mixture of Gaussians and elliptical slice sampling to perform updates on a variety of scales with self-adjusting step-sizes. Rather than updating each member of the

population in sequence, our approach splits the population into two groups and allows the members of each group to be updated in parallel.

Recent work on Hamiltonian Monte Carlo has attempted to reduce the tuning burden (Hoffman and Gelman, In press). A user friendly tool that combines this work with a software stack supporting Automatic Differentiation is under development (Stan Development Team, 2012). We feel that this alternative line of work demonstrates the interest in more practical MCMC algorithms applicable to a variety of continuous-valued parameter spaces and is very promising. Our complementary approach introduces simpler algorithms with fewer technical software requirements. In addition, our two-population approach to parallelization could be applied with whichever methods become dominant in the future.

6. Experiments. In this section, we compare Algorithm 3 with other parallel MCMC algorithms in three different settings in order to understand the usefulness of this algorithm in a variety of situations. First, we compare the performance of the algorithms on a sequence of increasingly ill-conditioned distributions. Second, we compare how each algorithm's performance scales with the number of cores used. Third, we compare how quickly the Markov chains mix on a variety of distributions.

6.1. *Samplers Considered.* We compare generalized elliptical slice sampling (GESS) with parallel versions of random-direction slice sampling (RDSS) (MacKay, 2003), coordinate-wise slice sampling (CWSS) (Neal, 2003), and Metropolis–Hastings (MH) Metropolis et al. (1953). These baselines run multiple Markov chains in parallel, and there is no interaction between chains. RDSS and CWSS are variants of slice sampling which differ in their choice of direction (a random direction versus a random axis-aligned direction) in which to sample. We also compare to a simple MH algorithm whose proposal distribution is a spherical Gaussian centered on the current state. Due to the speed of this approach, each sample we produce is the result of five MH updates so that the update steps for the various algorithms take comparable amounts of time. In each experiment, a tuning period is used to adjust the MH step size so that the acceptance ratio is as close as possible to the optimal value of 0.234 (Roberts and Rosenthal, 1998). This tuning is done independently for each chain. We do not compare our algorithm with algorithms such as Hamiltonian Monte Carlo which require that the user compute the gradient of the distribution and perform additional tuning.

6.2. *A Demonstration on Ill-Conditioned Distributions.* To demonstrate the ability of GESS to capture the shape of a distribution, we compare the parallel algorithms on a sequence of Gaussian distributions whose covariance matrices are increasingly ill-conditioned. We test on non-axis-aligned Gaussians in four dimensions. Each Gaussian has standard deviation 0.01 in three directions. In the fourth direction, the variance of the Gaussian ranges from 10^0 to 10^{11}. We sample each distribution using 115 parallel chains each for 10^5 iterations. We then use the samples to estimate the variance in the variable direction. The results are shown in Figure 4.

The results are clear: GESS provides an accurate estimate of the variance of the Gaussians even in the most skewed examples. RDSS, CWSS, and MH all begin to fail when the variance reaches around ten. This occurs because RDSS, CWSS, and MH are unable to account for the long, thin shape of the distribution. They take steps in uninformed directions, the vast majority of which lie outside the region of high density. As a result, the transition operators take very small steps, and successive states are highly correlated. In the case of GESS, the multivariate t approximation builds an understanding of the long, thin shape of the

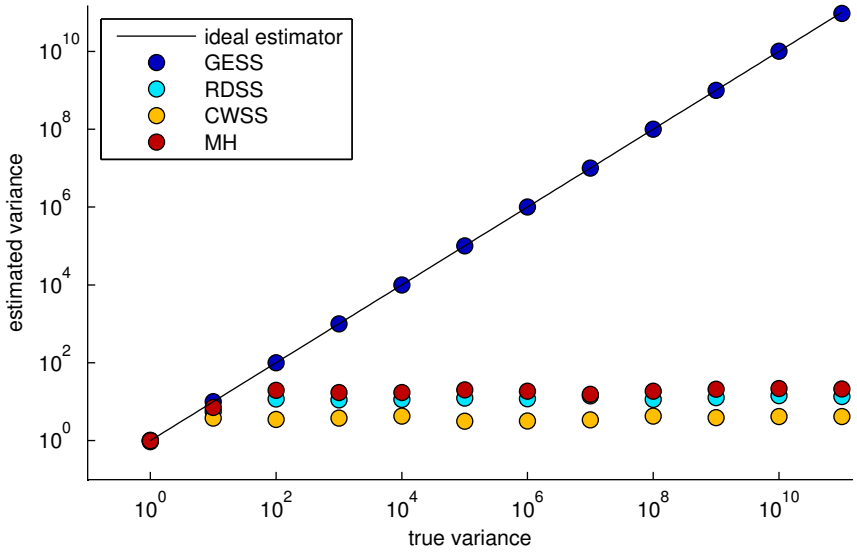


Fig 4: Four parallel MCMC algorithms are used to estimate the variance of a sequence of increasingly elongated non-axis-aligned multivariate Gaussian distributions. GESS gives an accurate estimate of the variance regardless of how skewed the distribution is because GESS is able to use shared information to dynamically builds an understanding of the distribution's shape into the transition operator. When the distribution becomes too skewed, RDSS, CWSS, and MH mix slowly and therefore give inaccurate estimates of the variance.

distribution into the transition operator. As a consequence, the Markov chain can take long steps along the length of the distribution and so convergence occurs much more rapidly. Such skewed distributions can arise as a result of the user not knowing the relative length scales of the parameters or as a result of a poor choice of parameterization. Therefore, the ability to perform well on such distributions is frequently relevant.

6.3. *Scaling the Number of Cores.* We wish to explore performance as a function of the number of cores. Here we test performance by evaluating estimates of known expectations under two different distributions. The first distribution is an axis-aligned Gaussian in forty dimensions. The standard deviations of the Gaussian are the integers from one to forty. We compare our four samplers on this distribution for approximately 10^5 iterations after 10^5 burn-in, and we thin by taking one of every hundred samples. We run this experiment with 64, 128, and 256 cores, and in each case we use one Markov chain per core. These experiments were run on an EC2 cluster with 10 nodes, each with two eight-core Intel Xeon E5-2670 CPUs. We use the resulting samples to estimate the mean of the distribution. Below we plot the magnitude of the error of the estimate as a function of wall-clock time. The results are displayed in Figure 5.

In this experiment, we find that the estimate generated by GESS converges the fastest, and we observe that GESS displays a steady decrease in error compared to the erratic behavior of the other algorithms. With each algorithm, increasing the number of parallel chains improved the rate of convergence.

The second distribution is a ten-dimensional funnel-shaped distribution described by Neal (2003). The first coordinate is distributed normally with mean zero and standard deviation three. Conditioned on the first coordinate v, the remaining coordinates are independent

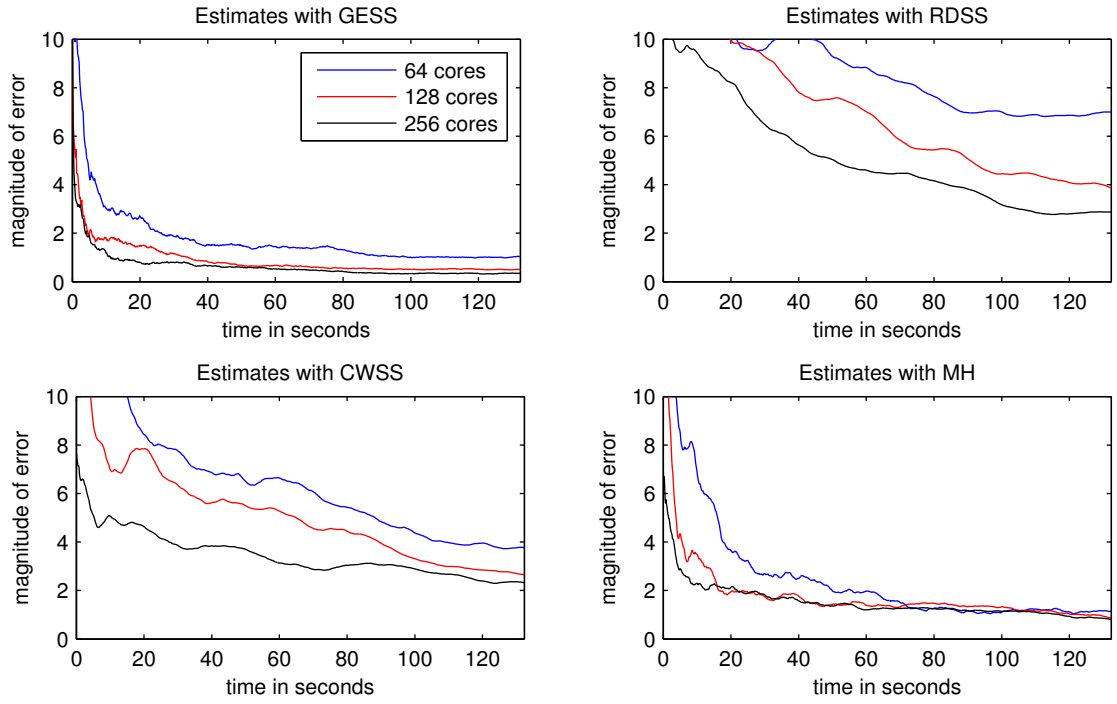


Fig 5: **Forty-dimensional Gaussian:** These plots show how performance scales as a function of the number of cores for the four different algorithms. Here, performance is quantified by using the simulated samples to estimate the mean of a forty-dimensional multivariate Gaussian distribution.

identically-distributed normal random variables with mean zero and variance e^v. In this experiment, we sample for approximately 10^6 iterations after 10^6 burn-in, and we thin by taking one of every thousand samples. We run this experiment with 32, 64, 128, and 256 cores, and in each case we use one Markov chain per core. Copying the test of Neal (2003), we compute the percentage of samples whose first coordinate is less than -5, and we plot the estimate as a function of wall-clock time. We know that the marginal distribution of the first coordinate is a Gaussian with mean zero and standard deviation three, so approximately 4.8% of samples should lie below this threshold. The results are displayed in Figure 6.

In this experiment, we find that the estimate generated by CWSS converges the fastest followed by GESS and then by RDSS. The estimate generated by MH is grossly inaccurate. The failure of MH in this setting has been explored by Neal (2003). When the algorithm converged, increasing the number of cores appeared to speed up the rate of convergence. The remarkable performance of CWSS can largely be attributed to the axis-aligned nature of the funnel distribution. To illustrate this, we run the same experiment on a rotated version of the same funnel. The results are shown in Figure 7. With this setup, CWSS and RDSS exhibit similar behavior, and GESS appears to converge the fastest.

6.4. *Comparing Mixing.* We empirically compare the mixing of the parallel MCMC samplers on five distributions. We quantify their mixing by comparing the effective number of samples produced by each method. This quantity can be approximated as the product of the number of chains with the effective number of samples from the product distribution.

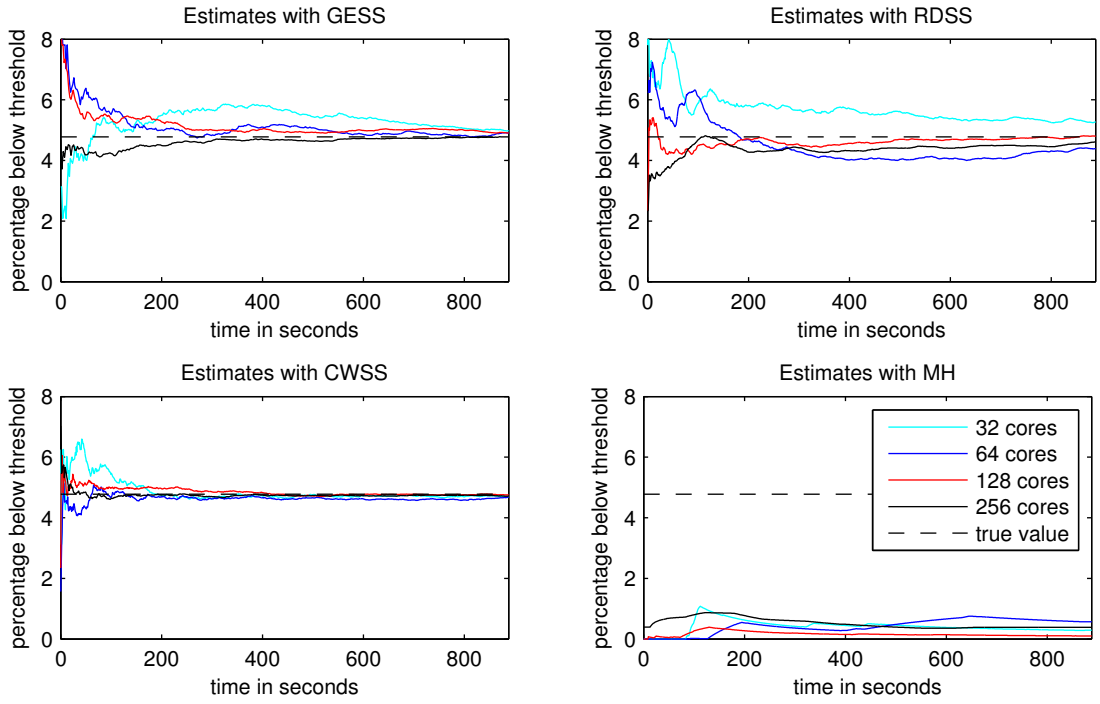


Fig 6: **Ten-dimensional funnel:** These plots show how performance scales as a function of the number of cores for the four different algorithms. Here, performance is quantified by using the simulated samples to estimate the percentage of samples whose first coordinate is less than -5 in a ten-dimensional funnel.

We estimate the effective number of samples from the product distribution by computing the effective number of samples from its sequence of log likelihoods. We compute effective sample size using R-CODA (Plummer et al., 2006), and we compare the results using two metrics: effective samples per second and effective samples per density function evaluation.

In each experiment, we run each algorithm with 115 parallel chains initialized from spherical Gaussian distributions. Unless otherwise noted, we burn in for 10^4 iterations and sample for 10^5 iterations. We run five trials for each experiment to estimate variability.

Figure 8 shows the average effective number of samples, with error bars, according to the two different metrics. Bars are omitted where the sequence of aggregate log likelihoods did not converge according to the Geweke convergence diagnostic (Geweke, 1992). We diagnose this using the tool from R-CODA (Plummer et al., 2006).

6.5. *Distributions.*

Funnel:. A ten-dimensional funnel-shaped distribution described in (Neal, 2003). The first coordinate is distributed normally with mean zero and variance nine. Conditioned on the first coordinate v, the remaining coordinates are independent identically-distributed normal random variables with mean zero and variance e^v.

Gaussian Mixture:. An eight-component mixture of Gaussians in eight dimensions. Each component is a spherical Gaussian with unit variance. The components are distributed uniformly at random along a hypercube with edge length four.

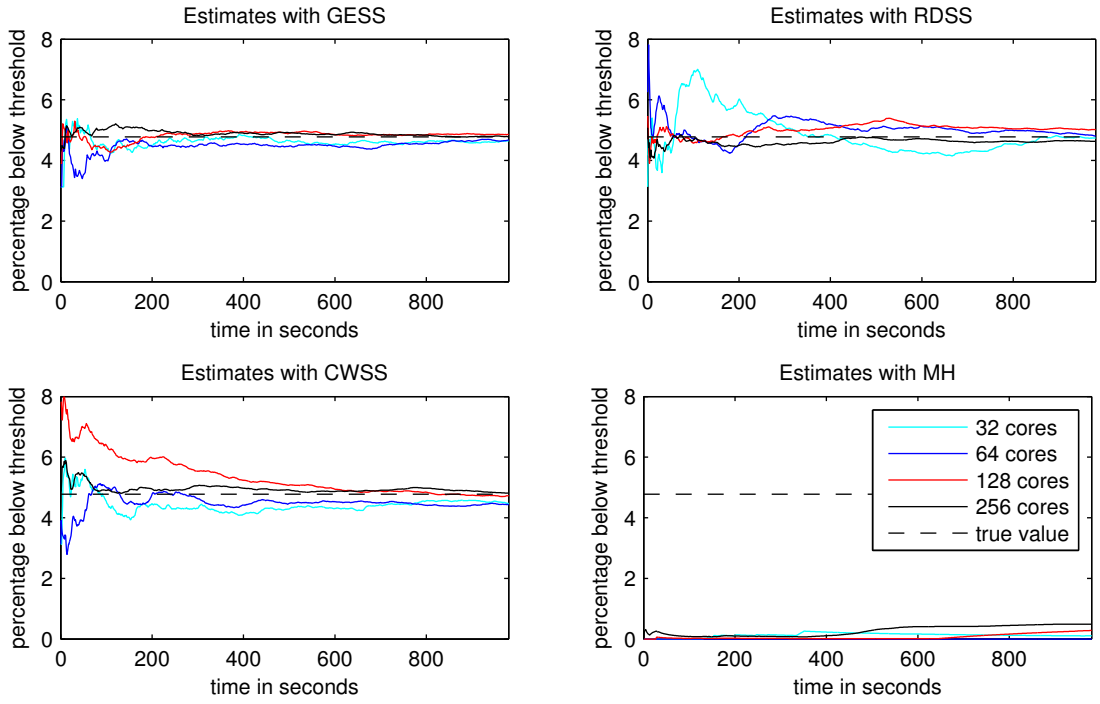


Fig 7: **Ten-dimensional rotated funnel:** These plots show how performance scales as a function of the number of cores for the four different algorithms. Here, performance is quantified by using the simulated samples to estimate the percentage of samples whose first coordinate is less than -5. The distribution is the ten-dimensional rotated funnel from Figure 6, but, but it has been rotated. This rotation should only affect CWSS.

Breast Cancer:. The posterior density of a linear logistic regression model for binary classification problem with thirty explanatory variables (thirty-one dimensions) using the Breast Cancer Wisconsin data set (Street et al., 1993). We scale the data so that each coordinate has unit variance.

German Credit:. The posterior density of a linear logistic regression model for binary classification problem with twenty-four explanatory variables (twenty-five dimensions) from the UCI repository (Frank and Asuncion, 2010). We scale the data so that each coordinate has unit variance.

Ionosphere:. The posterior density on covariance hyperparameters for Gaussian process regression applied to the Ionosphere data set (Sigillito et al., 1989). We use a squared exponential kernel with thirty-four length-scale hyperparameters and one hundred data points. We sample the posterior over hyperparameters. In this experiment, we burn-in for 10^4 iterations and sample for 10^4 iterations.

6.6. *Mixing Results.* In general, GESS sampled much more effectively than the other algorithms according to both metrics. However, GESS compared much more favorably when performance was measured by effective samples per evaluation than it did when performance was measured by effective samples per second. This is especially apparent in the funnel distribution given the extremely low cost of function evaluations. In this case, the construction of the multivariate t approximation and the overhead due to synchronization

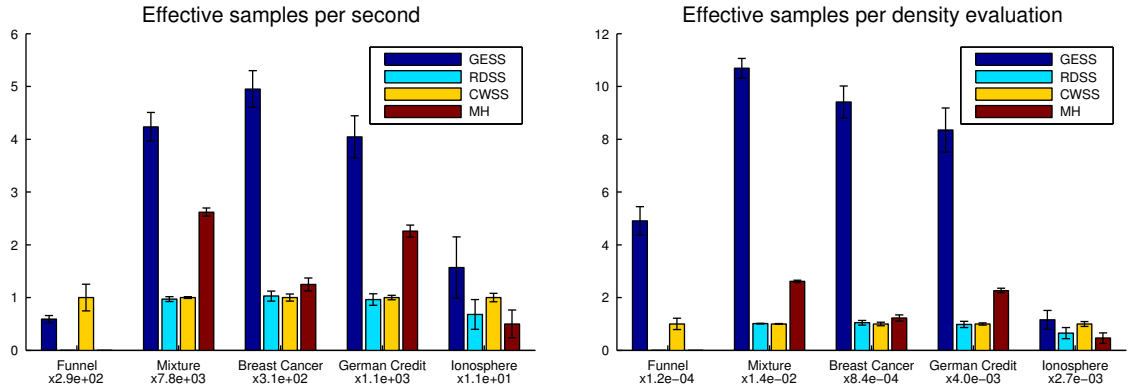


Fig 8: The results of experimental comparisons of four parallel MCMC methods on five distributions. Each figure shows five groups of bars, (one for each experiment) and the vertical axis shows the effective number of samples per unit cost. Error bars are included. Bars are omitted where the method failed to converge according to the Geweke diagnostic (Geweke, 1992). The costs are *per second* (left) and *per density function evaluation* (right). Mean and standard error for five runs are shown. Each group of bars has been rescaled for readability: the number beneath each group gives the effective samples corresponding to CWSS, which always has height one.

dominated the time cost.

These results show that a multivariate t approximation to the target distribution provides enough information to greatly speed up the mixing of the sampler. This improvement occurs on top of the performance gain from using parallelism.

7. Discussion. In this paper, we generalized elliptical slice sampling to handle arbitrary continuous distributions using a scale mixture of Gaussians to approximate the target distribution. We then showed that parallelism can be used to dynamically choose the parameters of the scale mixture of Gaussians in a way that encodes information about the shape of the target distribution into the transition operator. The result is Markov chain Monte Carlo algorithm with a number of desirable properties. In particular, it mixes well in the presence of strong dependence, it does not require hand tuning, and it can be parallelized over hundreds of cores.

We compared our algorithm to several other parallel MCMC algorithms in a variety of settings. We had several findings: generalized elliptical slice sampling (GESS) mixed more rapidly than the other algorithms on a variety of distributions, GESS accurately estimated the variances of highly-skewed distributions on which other algorithms failed, and GESS estimated expectations with increasing accuracy as the number of available cores increased.

One possible area of future work is reducing the overhead from the information sharing. In section 4.5 we remarked that the synchronization requirement leads to faster chains waiting for slower chains. There are a number of factors which contribute to the difference in speed from chain to chain. Most obviously, some chains may be running on faster machines. More subtly, the slice sampling procedure performs a variable number of function evaluations per update, and the average number of required updates may be a function of location. For instance, Markov chains whose current states lie in narrow portions of the distribution may require more function evaluations per update. In each situation, the chains with the rapid updates end up waiting for the chains with the slower updates, leaving some

processors idle. We imagine that a cleverly-engineered system would be able to account for the potentially different update speeds, perhaps by sending the chains in the narrower parts of the distribution to the faster machines or by allowing the slower chains to spawn multiple threads. Properly done, the performance gain in wall-clock time due to using GESS should approach the gain as measured by function evaluations.

In addition to using parallelism to distribute the computational load of MCMC, we saw that our algorithm was able to use information from the parallel chains to speed up mixing. One area of future work is extending the algorithm to take advantage of a greater number of cores. The magnitude of this performance gain depends on the accuracy of our multivariate t approximation, which will increase, to a point, as the number of available cores grows. However, there is a limit to how well a multivariate t distribution can approximate an arbitrary distribution. We chose to use the multivariate t distribution because it has the flexibility to capture the general allocation of probability mass of a given distribution, but it is too coarse to capture more complex features such as the locations of multiple modes. After some threshold, the approximation will not significantly improve. A more general approach would be to use a scale-location mixture of Gaussians, which could approximate any distribution arbitrarily closely. The idea of approximating the target distribution with a mixture of Gaussians has been explored by Ji and Schmidler (2010) in the context of adaptive Metropolis–Hastings. We leave it to future work to explore this more general setting.

Acknowledgements. We thank Eddie Kohler, Michael Gelbart, and Oren Rippel for valuable discussions. This work was funded by DARPA Young Faculty Award N66001-12-1-4219 and an Amazon AWS in Research grant.

References.

R. P. Adams, I. Murray, and D. J. MacKay. The Gaussian process density sampler. In *Advances in Neural Information Processing Systems 21*, 2009.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. Introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

R. Bradford and A. Thomas. Markov chain Monte Carlo methods for family trees using a parallel processor. *Statistics and Computing*, 6:67–75, 1996.

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.

S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195: 216–222, 1987.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

J. Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. *Bayesian Statistics*, 4:169–193, 1992.

W. R. Gilks, G. O. Roberts, and E. I. George. Adaptive direction sampling. *The Statistician*, 43(1):179–9, 1994. Special Issue: Conference on Practical Bayesian Statistics, 1992 (3).

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo. *Journal of the Royal Statistical Society B*, 73:1–37, 2011.

J. Goodman and J. Weare. Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science*, 5(1):65–80, 2010.

H. Haario, E. Saksman, and J. Tamminen. Componentwise adaptation for high dimensional MCMC. *Computational Statistics*, 20:265–273, 2005.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 1:97–109, 1970.

M. D. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, In press.

C. Ji and S. C. Schmidler. Adaptive Markov chain Monte Carlo for Bayesian variable selection. *Journal of Computational and Graphical Statistics*, 2010. to appear.

C. Liu and D. B. Rubin. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5:19–39, 1995.

J. S. Liu, F. Liang, and W. H. Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95:121–134, 2000.

D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

I. Murray, R. P. Adams, and D. J. MacKay. Elliptical slice sampling. *Journal of Machine Learning Research: W&CP*, 9:541–548, 2010.

R. M. Neal. Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. Technical Report 9508, Department of Statistics and Department of Computer Science, University of Toronto, 1995.

R. M. Neal. Regression and classification using Gaussian process priors. *Bayesian Statistics*, 6:475–501, 1998.

R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

R. M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman & Hall/CRC, 2011.

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006. URL http://CRAN.R-project.org/doc/Rnews/.

G. O. Roberts and J. S. Rosenthal. Markov chain Monte Carlo: Some practical implications of theoretical results. *Canadian Journal of Statistics*, 26:5–31, 1998.

G. O. Roberts and J. S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71, 2004.

G. O. Roberts and J. S. Rosenthal. Examples of adaptive MCMC. Technical report, Department of Statistics, University of Toronto, 2006.

J. S. Rosenthal. Parallel computing and Monte Carlo algorithms. *Far East Journal of Theoretical Statistics*, 4:207–236, 2000.

V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.

Stan Development Team. Stan: A C++ library for probability and sampling, version 1.0, 2012. URL http://mc-stan.org/.

W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, volume 1905, pages 861–870, 1993.

M. A. Suchard and A. Rambaut. Many-core algorithms for statistical phylogenetics. *Bioinformations*, 25: 1370–1376, 2009.

M. A. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West. Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 19:419–438, 2010.

D. Tarlow, R. Adams, and R. Zemel. Randomized optimum models for structured prediction. In *The 15th International Conference on Artificial Intelligence and Statistics*, 2012.

C. J. F. Ter Braak. A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces. *Statistical Computing*, 16:239–249, 2006.

L. Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22(4):1701–1728, 1994.

Robert Nishihara
Harvard University
Cambridge, MA 02138, USA
E-mail: robertnishihara@gmail.com

Iain Murray
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
E-mail: i.murray@ed.ac.uk

Ryan P. Adams
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, USA
E-mail: rpa@seas.harvard.edu